COMPUTER NETWORKS

SOCKETS CODING ASSIGNMENT

B180457EC	Ananthula Akshith

Q.Write a client and server using TCP socket programming. After the connection is established between client and server, the client should send a 'hello world' message to the server. The server should respond with a 'Hello, welcome' message. Use of socket programming is a must. Use Python or C++.

CODE:

Server.py:

```
import socket
def server_program():
    host = socket.gethostname()   # get the hostname
    port = 9123  # initiate with a random port no above 1024
    flag = 0     # initial value
    server_socket = socket.socket()  # get instance
    server_socket.bind((host, port))  # bind host address and port together
    server_socket.listen()    #no.of conections to listen
    conn, address = server_socket.accept()  # accept new connection
    print("Connection from: " + str(address))
    while True:
        # receive data stream. it won't accept data packet greater than 1024 bytes
        data = conn.recv(1024).decode()
        if not data:
            break          # if data is not received break
        print("from connected user: " + str(data))
        if data == "hello world":
          flag = 1
          data_send = "Hello, welcome"
        if flag == 0:
          data_send = input('>>> ')
        else:
          flag = 0
        conn.send(data_send.encode())  # send data to the client
    conn.close()  # close the connection

server_program()
```

Client.py:

```
import socket

def client_program():

    host = socket.gethostname()  # as both code is running on same pc

    port = 9123  # initiate with a random port no above 1024

    client_socket = socket.socket()  # instantiate

    client_socket.connect((host, port))  # connect to the server

    message = input(">>> ")  # take input

    while message.lower().strip() != 'bye':        #for ending the conversation!

        client_socket.send(message.encode())  # sends the message

        data = client_socket.recv(1024).decode()  # receive response

        print('Received from server: ' + data)  # show in terminal

        message = input(">>> ")  # again take input

    client_socket.close()  # close the connection


client_program()
```

Overall Explanation:

1) The code server.py starts to listen to incoming connection swhen it’s started and when the client.py starts running it sends request to server.py and then establishes the connection.
2) Here we are using the port 9123 since it’s not used as the port by any other application if not then the communication may interfere and get’s corrupted.
3) The command **socket.socket()** creates a socket So we create 2 sockets one from client and a server which enables to establish the connection.

Server Code Explanation:

1) The command **.bind((host,port))** is used to bind the tuple using that specific port and host name so that we can connect to the client using this tuple (This Tuple is necessary to establish the connection because it needs to know which computer to communicate and which port to send the message).
2) Now that we have succesfully binded the tuple we can now start listening to the incoming connections . For this we use the command **.listen()** to listen to the incoming connections.

3) Now when there is a incoming connection we use the command **.accept()** to accept the connection and establish the connection.

4) Now the connection is established we wait for the reply from the client and respond to the according to the message from the client by using the command **.send()** which sends the message to the client.

5) For recieving the message from client we use the command **.recv()**.

6) Now after sending and recieving the messages it's time to close the connection so we use **.close()** to close the connection.

Client Code Explanation:

1) The command **.connect((host,port))** is used to establish the connection using that specific port and host name.

2) Now that we have connected to the server we can send messages using the command **.send(message)** which sends the message to the server.

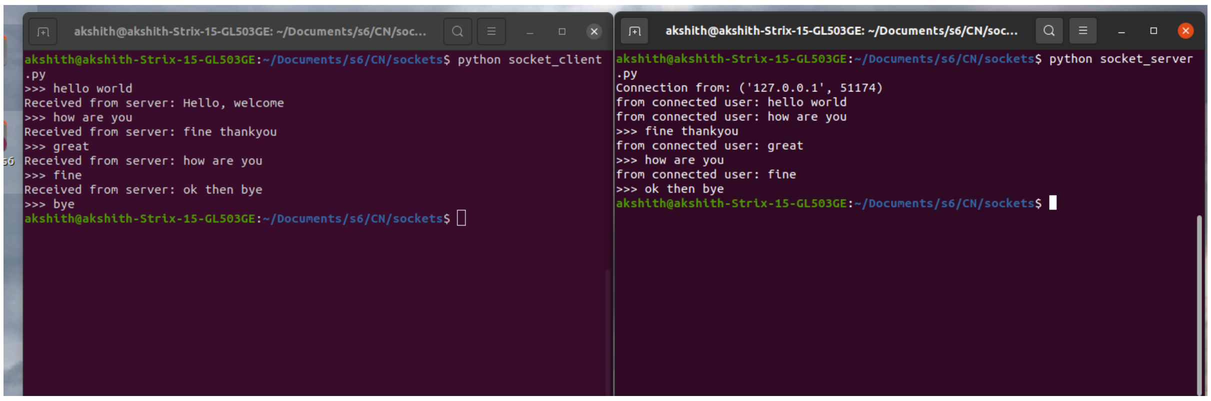
3) Now that we have send the message now it's time to recieve the reply from the server so we use the command **.recv()** which recieves the reply from the server.

4) Now after sending and recieving the messages it's time to close the connection so we use **.close()** to close the connection.

.decode() and .encode():

We use this to decode the message while recieving and encode the message while sending respectively.

Output:


akshith@akshith-Strix-15-GL503GE: ~/Documents/s6/CN/soc...
akshith@akshith-Strix-15-GL503GE: ~/Documents/s6/CN/soc...

```
akshith@akshith-Strix-15-GL503GE:~/Documents/s6/CN/sockets$ python socket_client
.py
>>> hello world
Received from server: Hello, welcome
>>> how are you
Received from server: fine thankyou
>>> great
Received from server: how are you
>>> fine
Received from server: ok then bye
>>> bye
akshith@akshith-Strix-15-GL503GE:~/Documents/s6/CN/sockets$
```

```
akshith@akshith-Strix-15-GL503GE:~/Documents/s6/CN/sockets$ python socket_server
.py
Connection from: ('127.0.0.1', 51174)
from connected user: hello world
from connected user: how are you
>>> fine thankyou
from connected user: great
>>> how are you
from connected user: fine
>>> ok then bye
akshith@akshith-Strix-15-GL503GE:~/Documents/s6/CN/sockets$
```

Result:

Successfully established the communication via sockets.